WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Friday, April 23, 2004

Hide?	Set Name	Query	Hit Count
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L24	l9 and l20	2
☐	L23	l5 and L20	0
☐	L22	l3 and L20	2
☐	L21	l2 and L20	12
☐	L20	l10 or l11 or l12 or l13 or l14 or l15 or l16 or l17 or l18 or L19	7463
☐	L19	707/102.ccls.	1808
☐	L18	707/100.ccls.	1444
☐	L17	707/3.ccls.	2470
☐	L16	707/1.ccls.	1607
☐	L15	717/121.ccls.	50
☐	L14	717/120.ccls.	93
☐	L13	717/106.ccls.	127
☐	L12	717/108.ccls.	224
☐	L11	713/100.ccls.	648
☐	L10	713/1.ccls.	959
☐	L9	(initializ$9 same (plug-in)).ab.	31
☐	L8	l6.clm.	1
☐	L7	l6.ab.	1
☐	L6	(initializ$9 same (plug-in) same list$4)	17
☐	L5	(generat$4 near3 (start-up) near2 sequence)	21
☐	L4	L3.ab.	8
☐	L3	L2 same computer	154
☐	L2	(start-up) near2 sequence	1424
☐	L1	(assembl$4 or construc$4) near2 (start-up) near2 sequence	2

END OF SEARCH HISTORY

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L24: Entry 2 of 2 File: USPT Dec 28, 1999

DOCUMENT-IDENTIFIER: US 6009520 A
TITLE: Method and apparatus standardizing use of non-volatile memory within a BIOS-ROM

Abstract Text (1):
A Basic Input-Output System (BIOS) includes a management and driver module adapted to accomplish editing functions for the BIOS. Plug-in modules are added to the BIOS by submitting the plug-ins to the driver module, which determines compatibility and available space for adding, and acts accordingly, adding a candidate module to the BIOS if space is available and the plug-in module is determined to be compatible with the BIOS and the driver module. Plug-ins can also be removed by action of the driver module, which also performs management functions in identifying and initializing resident plug-in modules.

Current US Original Classification (1):
713/1

Current US Cross Reference Classification (1):
713/100

First Hit Fwd Refs
End of Result Set

Generate Collection Print

L22: Entry 2 of 2 File: USPT Jul 28, 1992

DOCUMENT-IDENTIFIER: US 5134580 A
TITLE: Computer with capability to automatically initialize in a first operating system of choice and reinitialize in a second operating system without computer shutdown

Detailed Description Text (21):
This invention alters the flexible initialization system to enable a computer user who has finished a session on the DOS located in ROM to bring up the machine in an alternate operating system without the need to change the customized initialization from the preferred start-up sequence. Instead, the user need only place the alternate system on one of the system drives, (it could be on the fixed disk, for example) and then operate some indicia such as pressing the "alternate" key together with the "system request" key. The invention enables the user to retain the capability of thereafter starting the machine in the usual operating system of choice. This invention enables the user to go to games, which sometimes include their own rudimentary operating system, without turning the computer off.

Detailed Description Text (22):
The system has customizing bits which enable the user to temporarily change the start-up sequence by setting the "alternate system request" customizing bit (SR bit) through manually operated indicia such as the pressing of an appropriate combination of keys after bringing up the ROM shell screen shown in FIG. 2. When the combination of keys action is sensed, the SR bit is set and the computer is automatically reinitialized in the operating system or game found in the diskette drive, or on a fixed disk. During the reinitialization, the SR bit is reset so that future initializations will be in the expected operating system and not in the "alternate system request" mode. Since initializations clear memory, the writable permanent memory is used (CMOS) and the customization word set forth above includes the SR bits. The initializing routines to reset the SR bit are located in permanent read-only memory (ROM) as is the routine for recognizing the alternate system request key combination to set the SR bit.

Current US Original Classification (1):
713/1

Current US Cross Reference Classification (1):
713/100

☐ Generate Collection Print

L24: Entry 1 of 2 File: USPT Aug 6, 2002

DOCUMENT-IDENTIFIER: US 6430556 B1
TITLE: System and method for providing a query object development environment

Abstract Text (1):
A query object generator tool which generates interface definitions and code that implement a query object also generates a graphic user interface (GUI) for controlling the generator tool and plug-in objects, including a database schema access query object and test objects for allowing the GUI to operate with vendor-specific databases. The GUI is "customized" by the various plug-in objects. For example, the database schema access query object is designed specifically for a particular underlying database and retrieves "metadata" concerning the database schema. The retrieved metadata is then displayed as part of the graphic user interface to assist the user in constructing a query object. Test objects are also generated by the GUI in response to a user request. The test objects contain information that characterizes the query object for testing purposes. The information in the test objects is used with a test framework to install and initialize the query object. The test framework also uses the information in the test objects to customize part of the GUI in order to allow a user to view and interact with the installed query object. In particular, the customized GUI allows a developer to enter input parameters for a query directly from the interface and use the installed query object to perform a query with the input parameters. Results which are returned from the query are displayed on the interface.

Current US Cross Reference Classification (1):
707/102

Current US Cross Reference Classification (2):
717/108

L22: Entry 1 of 2 File: USPT Jun 4, 2002

DOCUMENT-IDENTIFIER: US 6401183 B1
TITLE: System and method for operating system independent storage management

Detailed Description Text (3):
Prior art computer devices include pre-determined start-up or boot sequences which are based on the firmware level of the device. These sequences ultimately load the run-time operating system for that device, which, in turn, controls the device.

Detailed Description Text (8):
During the start-up sequence of a prior art computer device having an MBR 10, after the boot program finishes the power-on self test (POST), the computer device loads the first sector of the first secondary storage device into main memory. The first sector is the location of prior art MBR 10. The computer device ensures that it has loaded the MBR 10 by cross-checking the MBR Signature Field 18 of the MBR 10 against a predetermined value. If the values match, then the computer device passes control to the IPL 12 of the MBR 10. The computer device then loads the IPL 12 into main memory along with whichever one partition is designated 15 as bootable in the partition table 14. If an operating system is stored within the bootable partition, then that operating system is booted into main memory and takes control of the computer system.

Detailed Description Text (20):
When activated, a computer device follows the start-up sequence generally illustrated in FIG. 11. The computer first performs its initial boot program at the Firmware Level 302, including the POST. By the means previously disclosed, the Storage Manager 306 is engaged prior to the run-time operating system being activated. The Storage Manager 306 thereby gains control of the computer device.

Current US Cross Reference Classification (2):
713/1

Current US Cross Reference Classification (3):
713/100

Generate Collection Print

L9: Entry 4 of 31 File: USPT Dec 28, 1999

DOCUMENT-IDENTIFIER: US 6009520 A
TITLE: Method and apparatus standardizing use of non-volatile memory within a BIOS-ROM

Abstract Text (1):
A Basic Input-Output System (BIOS) includes a management and driver module adapted to accomplish editing functions for the BIOS. Plug-in modules are added to the BIOS by submitting the plug-ins to the driver module, which determines compatibility and available space for adding, and acts accordingly, adding a candidate module to the BIOS if space is available and the plug-in module is determined to be compatible with the BIOS and the driver module. Plug-ins can also be removed by action of the driver module, which also performs management functions in identifying and initializing resident plug-in modules.

Generate Collection Print

L9: Entry 3 of 31 File: USPT Aug 6, 2002

DOCUMENT-IDENTIFIER: US 6430556 B1
TITLE: System and method for providing a query object development environment

Abstract Text (1):
A query object generator tool which generates interface definitions and code that implement a query object also generates a graphic user interface (GUI) for controlling the generator tool and plug-in objects, including a database schema access query object and test objects for allowing the GUI to operate with vendor-specific databases. The GUI is "customized" by the various plug-in objects. For example, the database schema access query object is designed specifically for a particular underlying database and retrieves "metadata" concerning the database schema. The retrieved metadata is then displayed as part of the graphic user interface to assist the user in constructing a query object. Test objects are also generated by the GUI in response to a user request. The test objects contain information that characterizes the query object for testing purposes. The information in the test objects is used with a test framework to install and initialize the query object. The test framework also uses the information in the test objects to customize part of the GUI in order to allow a user to view and interact with the installed query object. In particular, the customized GUI allows a developer to enter input parameters for a query directly from the interface and use the installed query object to perform a query with the input parameters. Results which are returned from the query are displayed on the interface.

L6: Entry 9 of 17 File: USPT Mar 12, 2002

DOCUMENT-IDENTIFIER: US 6356863 B1
TITLE: Virtual network file server

Detailed Description Text (65):
As can be seen from the above list, single-file file systems typically require between 2 and 6 functions to be implemented, and multiple-file file systems typically require between 9 and 12. When the file system plug-in is loaded, it is initialized (if mxINodeInitialize exists) and then mxINodeType is called to determine the type of the file system. If mxINodeType does not exist, the file system is assumed to be single-file. The plug-in manager then checks that all of the required functions are present. If not, the mxINodeCleanUp function is called (if it exists) and the module unloaded.

L6: Entry 10 of 17 File: USPT Jan 15, 2002

DOCUMENT-IDENTIFIER: US 6339771 B1
TITLE: Method and system for managing connections to a database management system

Detailed Description Text (3):
FIG. 1 is a block diagram of a system 10 for utilizing databases via the internet. The system 10 includes clients 12, 14, and 16. The clients 12, 14, and 16 are coupled with a server 20 via the internet 18. In order to access information through the server 20, the client 12, 14, or 16 sends a request (not shown) to the server 20. The server 20 includes a plug-in 28 and server software 30. The plug-in 28 may provide additional functionality to the server 20 through interactions with the server software 30 and other applications. The server software 30 includes run time structures also depicted in FIG. 1. The run time structures include a listener thread 32, a plurality of worker threads 34, and an initialization thread 36. The listener thread 32 listens for incoming requests. Each worker thread 34 aids in processing an incoming request. The initialization thread 36 initializes the server software 30. The server 20 is be coupled to databases 22, 24, and 26. Each of the databases 22, 24, and 26 holds information.

Detailed Description Text (11):
FIG. 4 depicts a block diagram of the run time structure 150 of the server 110 in accordance with the present invention. The run time structure 150 of the server 100 includes an initialization thread 122, a listener thread 124, and a plurality of worker threads 126. The listener thread 122 listens for incoming requests. Each worker thread 126 aids in processing an incoming request. The worker thread includes a service block 134. The service block is from code contained in the plug-in 130 and is invoked when a request, often termed a database request, utilizes the database 140. The initialization thread 122 initializes the server 110 for service block 134 processing. The initialization thread 122 is, therefore, responsible for generating the worker threads 126 and the listener thread 124. The initialization thread 122 includes a service initialization block 132 generated from code contained in the plug-in 130. The service initialization block 132 is used during initialization of the server 110. The service initialization block 132 creates a key during initialization of the server 110. The key is utilized in a preferred embodiment of the present invention and is further described below. Also in a preferred embodiment, the present invention is an internet connection server application programming interface ("ICAPI") solution.

L6: Entry 12 of 17 File: USPT Aug 29, 2000

DOCUMENT-IDENTIFIER: US 6112196 A
TITLE: Method and system for managing connections to a database management system by reusing connections to a database subsystem

Detailed Description Text (3):
FIG. 1 is a block diagram of a system 10 for utilizing databases via the internet. The system 10 includes clients 12, 14, and 16. The clients 12, 14, and 16 are coupled with a server 20 via the internet 18. In order to access information through the server 20, the client 12, 14, or 16 sends a request (not shown) to the server 20. The server 20 includes a plug-in 28 and server software 30. The plug-in 28 may provide additional functionality to the server 20 through interactions with the server software 30 and other applications. The server software 30 includes run time structures also depicted in FIG. 1. The run time structures include a listener thread 32, a plurality of worker threads 34, and an initialization thread 36. The listener thread 32 listens for incoming requests. Each worker thread 34 aids in processing an incoming request. The initialization thread 36 initializes the server software 30. The server 20 is be coupled to databases 22, 24, and 26. Each of the databases 22, 24, and 26 holds information.

Detailed Description Text (11):
FIG. 4 depicts a block diagram of the run time structure 150 of the server 110 in accordance with the present invention. The run time structure 150 of the server 100 includes an initialization thread 122, a listener thread 124, and a plurality of worker threads 126. The listener thread 122 listens for incoming requests. Each worker thread 126 aids in processing an incoming request. The worker thread includes a service block 134. The service block is from code contained in the plug-in 130 and is invoked when a request, often termed a database request, utilizes the database 140. The initialization thread 122 initializes the server 110 for service block 134 processing. The initialization thread 122 is, therefore, responsible for generating the worker threads 126 and the listener thread 124. The initialization thread 122 includes a service initialization block 132 generated from code contained in the plug-in 130. The service initialization block 132 is used during initialization of the server 110. The service initialization block 132 creates a key during initialization of the server 110. The key is utilized in a preferred embodiment of the present invention and is further described below. Also in a preferred embodiment, the present invention is an internet connection server application programming interface ("ICAPI") solution.

L6: Entry 4 of 17 File: USPT Feb 17, 2004

DOCUMENT-IDENTIFIER: US 6694510 B1
TITLE: Collection driver for collecting system data using record based requests with tag lists and pausing all but one thread of a computer system

Detailed Description Text (24):
When the an embodiment as illustrated in FIG. 1, 2, or 3, initializes for a particular target machine, any required network connection to the target machine is set up, the collection driver and communication service if required for that platform is started, and user interface 106, 216, or 340 of the debugger connects 500 (FIG. 5) through to the collection driver. If connection was unsuccessful, an error is reported 502 and operation ceases. If connection was successful, the build number and service pack number, or other version identifying information, is obtained 504 by the collection driver from the operating system of the target machine and returned to the user interface. A list of loaded executable modules on the target machine is also obtained 506, this list includes module names and checksums of the corresponding executable files. The build number and service pack number, or other version identifying information of the operating system, is used to ensure a compatible command plug-in 112, 222, or 324 is loaded, and used to locate symbol files in the library tree structure.

L6: Entry 5 of 17 File: USPT Oct 14, 2003

DOCUMENT-IDENTIFIER: US 6633923 B1
TITLE: Method and system for dynamic configuration of interceptors in a client-server environment

Detailed Description Text (240):
Plug-ins are listed in the configuration file and dynamically linked during ORB initialization. For each ORB instance created within a process, the plug-in maintains a per ORB state containing all data and references required for the ORB instance and it's specific configuration domain. This state is created by calls to the plug-in ART_PlugIn::ORB_init operation.

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L8: Entry 1 of 1 File: USPT Jul 30, 2002

DOCUMENT-IDENTIFIER: US 6427175 B1
TITLE: Method and apparatus for collaborative remote link management using sharable online bookmarks

CLAIMS:

36. A method of sharing access operations including viewing and editing of at least one bookmark node with an associated access level for each of said access operations by at least two identified web users including steps of: retrieving a first of said bookmark nodes from a network server located remotely from said identified web users; outputting said first of said bookmark nodes to a first of said identified web users; outputting said first bookmark node to a second of said identified web users; allowing performance of a first of said access operations of said first of said bookmark nodes by said first of said identified web users whenever said associated access level of said first access operation allows said first identified web user to perform said first access operation upon said first bookmark node; allowing performance of said first access operation of said first of said bookmark nodes by said second of said identified web users whenever said associated access level of said first access operation allows said second identified web user to perform said first access operation upon said first bookmark node; barring performance of said first access operation of said first bookmark node by said first identified web user whenever said associated access level of said first access operation bars said first identified web user from performing said first access operation upon said first bookmark node; barring performance of said first access operation of said first bookmark node by said second identified web user whenever said associated access level of said first access operation bars said second identified web user from performing said first access operation upon said first bookmark node; allowing performance of a second of said access operations of said first of said bookmark nodes by said first of said identified web users whenever said associated access level of said second access operation allows said first identified web user to perform said second access operation upon said first bookmark node; allowing performance of said second access operation of said first of said bookmark nodes by said second of said identified web users whenever said associated access level of said second access operation allows said second identified web user to perform said second access operation upon said first bookmark node; barring performance of said second access operation of said first bookmark node by said first identified web user whenever said associated access level of said second access operation bars said first identified web user from performing said second access operation upon said first bookmark node; barring performance of said second access operation of said first bookmark node by said second identified web user whenever said associated access level of said second access operation bars said second identified web user from performing said second access operation upon said first bookmark node; wherein at least one of said access operations is for editing the bookmark node; recognizing selection of a locally stored bookmark by said first identified web user, wherein selection of said locally stored bookmark calls said remote network server; identifying a web page being viewed by said first identified web user upon recognizing said selection of said locally stored bookmark; adding a link to said identified web page to said

first bookmark node stored on said remote network server; initiating a reporting mechanism installed locally; wherein said associated access level of said viewing operation by said first identified web user upon said first bookmark node allows said viewing operation by said first identified web user upon said first bookmark node whenever said associated editing access level of said editing operation by said first identified web user upon said first bookmark node allows performance of said editing operation upon said first bookmark node by said first identified web user; wherein each of said identified web users is further identified as a public user; wherein said first bookmark node access level of said first access operation may permit said public user to perform said access operation with said first bookmark node; wherein said first bookmark node is a bookmark folder; wherein said associated access level for each of said access operations by said identified web users of a first bookmark link contained in said first bookmark node folder has by default the same access levels for each of said access operations by said identified web users as for said first bookmark node folder; wherein said identified web users further contain an identified web user group including at least one of said identified web users wherein said associated access level for a first access operation of said first bookmark node is said identified group access level; wherein the associated access level for a first identified web user included in a first said identified web user group of said first bookmark node for each of said access operations is the same as the said first identified web user group access level; wherein said access operations further include adding a bookmark node with an associated access level by at least two identified web users; initializing a user account for an owner of said identified web user with a first bookmark folder on a server wherein said owner identified web user has editing access privileges and adding access privileges of said first bookmark folder on said server; providing a reporting mechanism for said owner identified web user to add a website address link as a bookmark link in said first bookmark folder on said server; running said reporting mechanism for said owner identified web user to add said website address link as said bookmark link in said first bookmark folder on said server; wherein said owner identified web user triggers said reporting mechanism while operating a web browser by calling said server to add said website address link as said bookmark link in said first bookmark folder on said server; wherein said providing of said reporting mechanism for said owner identified web user further includes downloading of a reporting program to a computer of said owner identified web user; installing of said reporting program on said owner identified web user computer; wherein triggering said reporting mechanism by said owner identified web user includes executing said reporting program on said owner identified web user computer which calls said server to add said web site address link in said first bookmark folder on said server; wherein said providing of said reporting mechanism for said owner identified web user further includes downloading of a plug-in to a web browser on a computer of said owner identified web user; installing of said plug-in to said web browser on said computer of said owner identified web user; wherein triggering said reporting mechanism by said owner identified web user includes running said plug-in from said web browser; wherein running said plug-in includes opening a history list of said web browser; extracting said website address link from said history list of said web browser; calling said server to add said website address link as a bookmark link in said first bookmark folder on said server; wherein said installing of said plug-in to said web browser on said computer of said owner identified web user includes adding a trigger bookmark to a local bookmark list of said web browser; wherein running said plug-in whenever said owner identified web user selects said trigger bookmark; wherein said installing of said plug-in to said web browser on said computer of said owner identified web user includes adding an icon on said owner identified web user computer; and wherein running said plug-in whenever said owner identified web user selects said icon.

First Hit

End of Result Set

Generate Collection Print

L7: Entry 1 of 1 File: DWPI Jul 6, 2002

DERWENT-ACC-NO: 2003-036825
DERWENT-WEEK: 200303
COPYRIGHT 2004 DERWENT INFORMATION LTD

TITLE: Method of supplying mobile multimedia communication using multimedia streaming format

Basic Abstract Text (2):
DETAILED DESCRIPTION - A client starts rendering(S1), and lists-up plug-in CODECs (S2). The client downloads an RSF(Rostic Streaming Format) stream(S3), and searches control data for initializing the CODECs(S4). The client checks whether the CODECs exist(S5). If so, the client checks a version of the CODECs(S6). If the CODECs do not exist, the client displays an error message and completes the rendering(S7). The client opens the CODECs after checking the version(S8), and decodes an audio frame as decoding a video frame(S9). The client closes the CODECs(S10), and completes the rendering(S11).

L6: Entry 2 of 17 File: USPT Mar 16, 2004

DOCUMENT-IDENTIFIER: US 6708189 B1
TITLE: Computer file transfer system

Detailed Description Text (37):
Upon creation of a script, the script processor 46 may enter an initialize routine (FIG. 4F). The processor may initialize core components 126, such as its own internal registers and load preferences 128, such as identifiers of hard drives and other storage devices. The processor 46 may search 130 for the locations of plug-ins and create 132 a plug-in option list. Scripts may, in turn, be loaded 134.

L3: Entry 1 of 4 File: USPT Feb 20, 2001

DOCUMENT-IDENTIFIER: US 6191964 B1
TITLE: Circuit and method for controlling a synchronous rectifier converter

Detailed Description Text (27):
Turning now to FIG. 7, illustrated is a timing diagram 700 of a start-up sequence for a plurality of converters operating in a parallel forced load-sharing converter circuit (or system). The synchronous rectifiers of the converters are enabled as a function of time to prevent reverse power flow for the converter circuit. This technique may be especially effective during start-up, where a synchronous rectifier is particularly susceptible to reverse power flow. The timing diagram 700 illustrates a typical start-up sequence that could occur during, for instance, a hot plug-in condition. A voltage curve Vout1 represents the voltage level of one or more power converters already operating in the converter system. A voltage curve Vout2 represents the voltage level of an additional power converter being added to the converter system. At a time t0, the additional power converter is enabled and begins its soft start sequence. At a time t1, the converter system reaches steady state, with all power converters at the same voltage level. Note that the voltage levels Vout1, Vout2 may represent the relative duty ratios of the converters in the system, since the duty ratio is proportional to the voltage in the converters. The is additional converter is susceptible to reverse power flow for the period before the time t1. Therefore, it is desirable to disable the synchronous rectifier(s) for the period up to time t1.